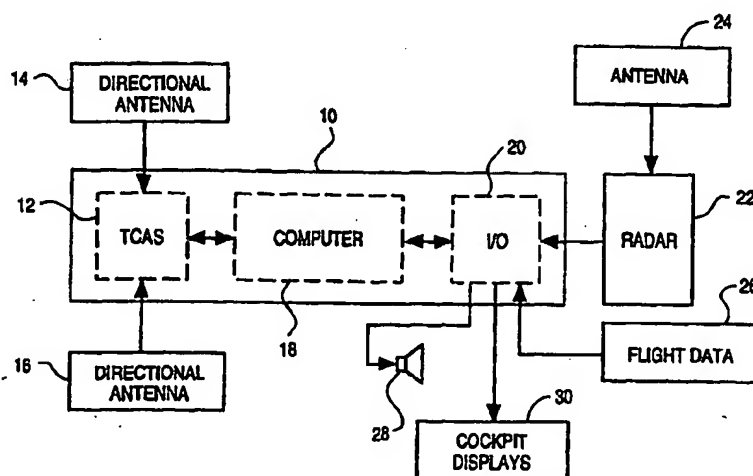




INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification ⁶ : G06F	A2	(11) International Publication Number: WO 99/36847 (43) International Publication Date: 22 July 1999 (22.07.99)
(21) International Application Number: PCT/US99/01221 (22) International Filing Date: 20 January 1999 (20.01.99) (30) Priority Data: 60/071,914 20 January 1998 (20.01.98) US Not furnished 19 January 1999 (19.01.99) US (71) Applicant: ALLIEDSIGNAL INC. [US/US]; 101 Columbia Road, P.O. Box 2245, Morristown, NJ 07962-2245 (US). (72) Inventors: MCELROY, James, J.; 17428 - S.E. 60th Street, Bellevue, WA 98006-5906 (US). JOHNSON, Clark; 10909 Avondale Road, N.E. #S-173, Redmond, WA 98052 (US). (74) Agents: CRÏSS, Roger, H. et al.; AlliedSignal Inc., Law dept. (R. Fels), 101 Columbia Road, P.O. Box 2245, Morristown, NJ 07962-2245 (US).		(81) Designated States: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE). Published <i>Without international search report and to be republished upon receipt of that report.</i>

(54) Title: FAULT TOLERANT COMPUTING SYSTEM USING INSTRUCTION COUNTING



(57) Abstract

In order to provide a microprocessor based fault tolerant computing system, hardware counters or event monitors that are normally included on the microprocessor chips are used to count application instructions that are being executed by the microprocessors. By counting the instructions and preempting the execution of the application program after a predetermined number of instructions have been executed, it is possible to cause the application programs to execute in congruent frames so that results from the application can be checked at congruent points of their execution. If the results do not match, then the program can be terminated or if a number of microprocessors are being used, the results can be voted on and the ones that match can be used in further computation by the system.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon			PT	Portugal		
CN	China	KR	Republic of Korea	RO	Romania		
CU	Cuba	KZ	Kazakhstan	RU	Russian Federation		
CZ	Czech Republic	LC	Saint Lucia	SD	Sudan		
DE	Germany	LI	Liechtenstein	SE	Sweden		
DK	Denmark	LK	Sri Lanka	SG	Singapore		
EE	Estonia	LR	Liberia				

5

Fault Tolerant Computing System Using Instruction Counting

FIELD OF THE INVENTION

10 The invention relates to fault tolerant computing systems and in particular to fault tolerant systems where application programs are synchronized at the processor level.

BACKGROUND OF THE INVENTION

15 Systems to achieve fault tolerant computing have historically required special purpose hardware connecting several processors or else software design restrictions that greatly complicate the design and implementation of application software. Examples of such systems are described in U.S. Patent Nos. 4,937,741, 5,388,242, 5,491,787 5,537,655 and 5,588,111.

20 There is a need for a low cost but high performance fault tolerant computing system that does not greatly increase the difficulty of application software design. It is generally recognized that there is a need to employ digital computers in applications in which improper operation could have severe consequences. For example, a sophisticated flight hazard warning system has been developed for aircraft
25 which utilizes a number of independent warning systems including a ground proximity warning system, a wind shear detection system and a collision avoidance system. This particular system is generally described in U.S. Patent Application Serial Number 08/847,328, filed April 23, 1997 and entitled: "Integrated Hazard Avoidance System", and is incorporated herein by reference. In the preferred embodiment
30 described therein, a central computer, which may include multiple processors for redundancy, receives via various input/output (I/O) modules various types of flight data useful for anticipating and warning of hazardous flight conditions. Such information may include but is not limited to: barometric altitude, radio altitude, roll and pitch, airspeed, flap setting, gear position, and navigation data. This information
35 is communicated to the central computer via a data bus.

For such an integrated warning system to provide warnings with a high degree of integrity, the data operated upon and instructions issued by the central computer

5 must be accurate. A bus architecture to transfer data between each of the I/O modules in an orderly manner must therefore exist. Data placed on the bus must also be accurate and without error. Also, it is important to ensure, to the extent possible, that the individual systems execute the warning programs correctly.

There have been various approaches to solving these problems. For example
10 such a system is described in ARINC Specification 659 entitled Backplane Data Bus published on December 27, 1993 by Aeronautical Radio, Inc. In this system the bus includes four data lines and has a pair of Bus Interface Units("BIU")for each processor or node on the data system where each BIU is connected to two data lines in the bus. Data is transferred according to a time schedule contained in a table memory
15 associated with each BIU. The tables define the length of time windows on the bus and contain the source and destination addresses in the processor memory for each message transmitted on the bus. These types of systems also use for some applications two processors that operate in a lock-step arrangement with additional logic provided to cross-compare the activity of the two processors. The two processors, each with its
20 own memory, execute identical copies of a software application in exact synchrony. This approach usually requires that the two processors must be driven by clock signals that are synchronized.

Although such systems have high data integrity and provide for fault tolerant operation, they have a number of disadvantages. For example the use of tables having
25 data source and destination addresses for each application program in the processor memory makes it difficult to reprogram the system for new applications because each table in the system must be reprogrammed. In addition, the use of two processor operating in lock-step reduces the flexibility of the system since it is not possible to run two different programs on the processors at the same time.

30

SUMMARY OF THE INVENTION

This invention provides a way of using hardware facilities that are part of commercially available microprocessors together with control software to implement
35 a fault tolerant computing system. Using the technique of this invention, a robust fault-tolerant computing system can be built. Application software that executes on

5 the system can remain simple because it does not need to be aware of the measures taken to achieve the fault-tolerant characteristics of the system, that is, no special redundancy management code is built into application code. The redundancy management code is entirely at the operating system level. In addition, the application software does not need to adhere to restrictive design rules to allow the system's fault
10 detection and containment mechanisms to work. The invention thus provides a way to separate the concerns of fault tolerance mechanisms and application logic. This makes it much easier and therefore less expensive to build robust fault tolerant computing systems.

This invention has commercial value because it allows strong and robust fault
15 tolerant computing systems to be built with low cost commercial off the shelf components. For example, by using counters or event monitors that are built into the microprocessor chips, it is possible to count the instructions being executed in an application program so as to cause the application programs to execute in congruent frames. Therefore, systems built using this technology will have a substantial
20 advantage over systems built with fault tolerant architectures that require custom electronics, custom integrated circuits or tricky and expensive application software design techniques.

This invention is also valuable because it uses technology that will be enhanced and extended as part of the natural growth path of microprocessor
25 technology. Future microprocessors and microcontrollers now on the drawing boards having hardware that can be used to monitor the execution of application programs will almost certainly increase the advantage of this approach to fault tolerant systems.

BRIEF DESCRIPTION OF THE DRAWINGS

30 Fig. 1 is a block diagram of fault tolerant computing system according to the invention;

Fig. 2 is a logic diagram of a data input function of the system of Fig. 1 according to the invention;

Fig. 3 is a logic diagram illustrating data verification in the system of Fig. 1;
35 and

5 Fig. 4 is a timing diagram illustrating loose synchronization of microprocessors according to the invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

10 The following description of the preferred embodiment of the invention utilizes as an illustrative example the MPC860 microcontroller manufactured by Motorola although the invention is not restricted to the use of this particular microprocessor.

 This microcontroller includes a 32-bit microprocessor which is a variant of the
15 IBM/Motorola PowerPC architecture together with a second processor called the Communications Controller (CP). The component also includes 8 serial I/O adapters integrated on-chip with the processor; of these, four are of similar type and are referred to as Serial Communications Controllers (SCCs).

 The PowerPC part of the component includes a Memory Management Unit
20 (MMU) that supports page level memory allocation and relocation. The MMU is able to treat physical memory as a range of pages of 4096 bytes. Each page is simply a range of addresses of 4096 bytes length. When the MMU is enabled, the code that is executing in the processor identifies an address space consisting of pages having locations which are controlled by the MMU. The MMU is capable of making various
25 pages accessible or inaccessible to the executing code. In addition, the MMU may alter the address ranges where the executing code "sees" each page of physical memory (relocation).

 These features allow the implementation of a system in which applications execute in a user mode and an operating system or executive executes in a supervisor
30 mode. The executive software is able to cause the applications to execute in a controlled environment and also can provide services to the applications.

 A feature of the invention involves the use of a processor hardware feature that can preempt (interrupt) executing application code after a precise number of instructions have been executed rather than the usual method of preempting executing
35 code after a predetermined amount of time has elapsed. As explained below, this

5 capability allows essentially unrestricted application software architectures to be used with the system. The use of this feature has very significant advantages in implementing fault tolerant computing systems.

Fig. 1 shows an example of a fault tolerant computing system 10 having a group of six microprocessors 12A-F, such as the MPC 860, that can be programmed with appropriate software that makes use of the invention. Each of the
10 microprocessors 12A-F is able to communicate with all others by means of a group of serial buses indicated at 14, using a set of Serial Communications Controllers (SCCs) 16A-D that are incorporated on the microprocessors 12A-F with the microprocessor cores. The three microprocessors 12A-C to the left in Fig. 1 are also connected to
15 external devices by means of an I/O circuit 18A-C as needed for the application. The microprocessors 12A-C are thus able to take care of the input/output functions of the system 10 while the microprocessors 12D-F on the right of Fig. 1 are able to perform processing duties, obtaining input and returning output to the microprocessors 12A-C on the left.

20 In this embodiment, each of the microprocessor 12A-F drives only one of the serial busses 14, but is able to receive from all of the serial busses 14. Also, each of the microprocessors 12A-F has only four SCCs 16A-D. Therefore, the microprocessors 12A-F are not capable of obtaining data simultaneously from more than four of the microprocessors 12A-F. For this reason, a set of four multiplexer
25 circuits 20A-D are added external to each of the microprocessors 12A-F to enable them to obtain data from all the microprocessors 12A-F in a time-sequenced manner. Software at the executive level in each of the microprocessor 12A-F is used to synchronize and sequence the communications between the microprocessors 12A-F. With 4-to-1 multiplexers 20A-D as shown in Fig. 1, up to sixteen microprocessors
30 could be used in a system.

Fig. 2 illustrates in logical block diagram form three data input processing functions as might be implemented on the microprocessors 12A-C that have access to I/O signals from the I/O circuits 18A-C shown in Fig. 1. Each of these functions periodically sample signals from various sensors as indicated by a set of blocks 22A-
35 C. These signals can be from redundant sensors but, in general, they need not be identical or even synchronized with each other. After sampling, the signals are sent

5 for rule based voting as indicated by a set of function blocks 24A-C. The function indicated by the blocks 24A-C might be implemented in software executing on the same microprocessor 12A-C as the data input function in the blocks 22A-C or on different microprocessors in the system 10.

10 After the data exchange, each of the rule based voting functions 24A-C has access to all of the input signals from blocks 22A-C, both its own and those from the other microprocessors 12A, B or C. Rules are then applied to the available signal samples to determine which ones to select and how to make the best use of the available data. This function depends on the details of the input data and the sensors from which the data is obtained and would readily be apparent to those of ordinary
15 skill in the art of data verification.

When the rule based voting functions 24A-C have finished processing a frame of data, they will have separately produced, in the absence of faults, three independent versions of the input data. The three versions of the input data in this embodiment will be loosely synchronized and bit-for-bit identical. This property makes fault
20 containment much easier in the subsequent processing.

The next processing steps occur at some point between the output of the rule based voting identified by the blocks 24A-C and three identical application programs represented by a set of blocks 26A-C in Fig. 2 that make use of the incoming data from the sensors via the blocks 22A-C.

25 Continued operation of the system 10 is explained in connection with the logic diagram of Fig. 3. After the input processing described above, the manner in which the rest of the system 10 in this embodiment of the invention will detect faults and continue operation is as follows. As shown in Fig. 3, a transmitting processor, for example microprocessor 12A, executes one instance of the input processing function.
30 This microprocessor 12A produces a sequence of output data sets that are then transmitted via that microprocessor's SCC 16D to the serial output bus 14. Also shown is a receiving processor, for example the microprocessor 12D, that is the host of one of the redundant instances of the receiving application 26A. Not shown in Fig. 3 are the other transmitting microprocessors 12B and C and the other receiving
35 microprocessors 12E and F, but it should be understood that operation of these elements of the system 10 will be similar to the ones shown in this figure.

5 Here the receiving microprocessor 12D obtains the data from the transmitting microprocessor 12A on one of its serial input buses 14. At the same time, it receives data from several other transmitting processors 12B and C. The system 10 is preferably set up so that the data obtained from the several processors is loosely synchronized and congruent as long as there are no hardware faults in the system.

10 With this arrangement, it becomes possible to use voting at the receiving microprocessors such as 12D-F to handle faults in the incoming data.

 Voting consists of examining the incoming data and checking it to determine if all copies match exactly as indicated by a set of blocks 28A-C shown in Figs. 2 and 3. If all the incoming data matches, then any version of the data can be used. If only two

15 versions of the data match, then either of the matching versions can be used and the non-matching data is discarded as illustrated at 30 of Fig. 3. If there are no matches then the voter will be unable to determine which, if any, of the data sources should be trusted but for this to occur more than one failure must have occurred. Thus, the architecture of the system 10 can effectively operate with any single failure in the data

20 input system and while providing loosely synchronized congruent data for use by subsequent application programs such as application programs 26A-C.

 One of the most significant features of the invention permits the software applications such as 26A-C to be executed in the microprocessors 12D-F in such a way that it becomes possible to check the execution of those applications 12A-C, and

25 to reject any that produce faulty outputs, without the necessity of designing or implementing application programs with fault tolerance in mind. It should be noted that the assumption in this description of the invention is that the faulty output would occur because of a hardware processing fault not a software fault. As discussed below, system verification is accomplished by comparing the results from one

30 instance of the application executing on one processor with the results from another identical instance of the application executing on another processor.

 The voter 28A now selects any one of the input message blocks that has the maximum number of matches in agreement with other message blocks. In the example shown in Fig. 3, a second 32 or a third message block 34 might be selected

35 because the first message block 30 contains an error.

- 5 To summarize the data selection for the application programs 12A-C, the following steps are taken in the preferred embodiment of the invention:
1. Incoming messages are received from multiple sources and stored.
 2. A selection from among the various inputs was made by software external to the Receiving Application. This software is controlled by, or is part of, the
10 operating system software in the preferred implementation.
 3. The receiving application is not aware of the message selection made and does not take part in any way in accomplishing voting or redundancy management on its own. Therefore, its logic is not made more complicated by redundancy management issues.
 - 15 4. Testing and proving such a system, such as certification for use on an aircraft, is made simpler and less expensive because the fault tolerant aspects of the system can be understood, tested and proven independently of the complexities of the application software.

It should be noted that data message from the transmitting processors 12A-C
20 need not be tightly synchronized. However, it is desirable in this particular embodiment that the voters 28A-C wait until all the messages have arrived or should have arrived before voting; therefore loose synchronization is needed. For example, it can be required that non-failed data sources deliver their data messages before a predetermined time has expired.

25 In the fault tolerant system of the invention, it is desirable to insure that new data is presented to the several redundant application programs 26A-C only at points in the programs' execution that is the same on all processors. In other words, it is desirable to synchronize the operation of the microprocessors 12D-F during the processing of the application programs. As described above, this can be done by one of several
30 methods. For example:

1. Restrict the software design of the application so that data reading only occurs at points in the program execution that can be controlled relative to the points where the program is preempted. In this case, the software application designer must understand and take into account the method of obtaining fault tolerance.

- 5 2. Restrict the software design of the application so that it executes in "frames" or "steps" that run to completion each time the application is started. No new data is provided to the program once it starts executing. New data is provided only between steps.
3. Add hardware to the system that forces the processors to run in "lock step" on an
10 instruction by instruction basis. This, of course, provides for a system that always provides data to the several applications at the same points in program execution even if data is continuously arriving.

However, each of these synchronization approaches has drawbacks that can add substantial complexity to either the hardware or the software design.

- 15 To overcome these disadvantages, hardware features that are present on certain types of microprocessors such as the MPC860, as described below, can be used to maintain loose synchronism of the microprocessors while they independently execute the same application program. This method of maintaining the congruence of redundant applications on several loosely synchronized processors can be summarized
20 as follows:

1. Set up congruent input data for each of the redundant programs on the several microprocessors as described above.
2. Start execution of the application program in each microprocessor at known points either at a standard start up point or at the instruction following a congruent
25 preemption which preferably will be made to occur after the same instruction on all the microprocessors.
3. Count executed instructions from the start point, and when a suitable number of instructions have been executed, preempt or interrupt the application program on the microprocessor. This is defined as a "congruent preemption".

- 30 This method is similar to the run to completion frames mentioned above, but the frames are in effect created at run time. There is no need for the application software designer to know or care how the fault tolerant characteristics of the system are achieved.

5 Fig. 4 provides, in a timing diagram form, an illustration of the two
microprocessors 12D and 12E executing a common application program utilizing
congruent preemption under the control of an operating system according to the
preferred embodiment of the invention. In this example, the executive or operating
system of both microprocessors 12D and 12E are running at points C1 and D1
10 respectively. At approximately the same time, C2 and D2, the executive causes the
application programs 26A and 26B to begin to execute. After a predetermined count
of application instruction completions in each of the microprocessors 12D and 12E,
each executive generates a congruent preemption to halt the execution of the
application programs as shown at C3 and D3. At this point, the system 10 can
15 compare the results of the two application programs or if the output of the programs
are to be transmitted to another system, the results can be voted on to insure an
accurate output. Then after the executives in each of the microprocessors 12D and
12E complete their tasks, they initiate a resumption in the application programs 26A
and 26B at points C4 and D4. This process continues indefinitely or until the
20 application programs 26A and 26B are completed. In this manner the applications,
26A - 26C can be made to execute in congruent frames.

It should be understood that with this method of counting instruction
executions, or instruction completions, the application programs will, in general, be
interrupted at somewhat different times on the several microprocessors, but the last
25 instruction to complete will correspond on each processor at each congruent
preemption. Once this has been accomplished, it is easy to guarantee that all
application programs will start again on a corresponding instruction, regardless of
how they are programmed.

This method makes use of built-in hardware on the microprocessor chip that is
30 usually used for other purposes such as debugging the chip to count the instructions
executed in the application program. A number of commercially available
microprocessors including the Motorola MPC860, the MPC823, the 604e and the 750
contain suitable hardware to perform this function. The MPC860 for example has a
subsystem that was designed as an aid to software development and debugging that is
35 called "Development Support" by Motorola. It includes eight internal comparators
that can detect various events that occur during instruction execution. It also includes

5 two 16-bit counters that are capable of counting events. Use of this subsystem for counting instruction completions involves setting up the comparators to detect instructions executed within a desired address range, for instance the address range where the application being monitored is located, and setting up a counter to count the number of events detected. When a predetermined number of counts is reached, the
10 processor will "trap" to an executive routine.

The models 604e and the 750, contain a different mechanism, but one that can also be put to the same use. In these processors there is a subsystem referred to as the "Performance Monitor". This facility was designed into the processors to provide the ability to monitor and count predefined events such as processor clocks, misses in the
15 instruction cache, data cache, or L2 cache, the types of instructions dispatched, mispredicted branches and other events. In particular, it is possible to set up the event monitor system to take a "Performance Monitor Interrupt" after a predetermined number of application program instruction completions.

Other microprocessors have similar capabilities. Also, as microprocessors
20 become more complex and as the amount of logic on the processors increases, the need for features to assist with debug and performance measurement increases. For these reasons, it is likely that in the future more and more microprocessors will include a mechanism capable of counting instruction completions. Such microprocessors may then be used in fault tolerant architectures of this type.

25 It will be understood that there are a variety of ways of implementing the invention described above. The system disclosed in the U.S. Provisional Patent Application Serial No. 60/071,914 filed on 01/20/98 and assigned to the assignee of this application and hereby incorporated by reference in this application, provides an example of such an implementation.

30 Thus, by using instruction completion counters in microprocessors instead of time for preempting the execution of an application program in combination with organizing input data into packages that are congruent across the several microprocessors that will be executing the redundant instances of the application program, it is possible to create a fault tolerant system which is comparatively
35 inexpensive as well as transparent to application programmers.

5 We claim:

1. A fault tolerant computing system comprising:
a plurality of microprocessors each having a substantially identical application
program and each having an executive program;
10 data input means for applying the same data to said microprocessors;
count means located in each of said microprocessors for counting each
instruction of said application program executed by said microprocessors;
start means associated with said executive program for starting said
application program on each of said microprocessors at approximately the same time;
15 preemption means associated with said executive program and responsive to
said count means for halting said application program on each of said microprocessors
after a predetermined number of said instructions of said application program have
been executed;
comparison means for comparing the results of said application programs after
20 said predetermined number of instructions have been executed; and
resumption means associated with said executive program for resuming the
execution of said application program on each of said microprocessors.
2. The system of Claim 1 wherein said count means includes a counter in each of
said microprocessors for counting said application instructions executed by said
25 microprocessors.
3. The system of Claim 1 wherein said count means includes an
event monitor in each of said microprocessors for counting said application
instructions executed by said microprocessors.
4. The system of Claim 1 wherein said preemption means and resumption means
30 cooperate to complete the execution of said application program on each of said
microprocessors.
5. The system of Claim 4 wherein there are at least three of said microprocessors
and additionally including voting means operatively associated with said comparison
means for selecting matching results of said application programs.
- 35 6. The system of Claim 4 wherein said data input means includes voting means
associated with each of said microprocessors for selecting accurate data from a
plurality of data sources.

5

7. A method for fault tolerant computing utilizing a plurality of microprocessors each having a hardware counter comprising the steps of:

inputting the same application program into each of the microprocessors;

inputting congruent data into each of the microprocessors;

10 starting execution of said application program in each of said microprocessors at approximately the same time;

counting the number of the instructions of said application program executed by the microprocessors using the hardware counter in the microprocessors;

15 halting said execution of said application program in each of the microprocessors after a predetermined number of said instructions have been executed as determined by the counter;

comparing the results of said application program from each of said microprocessors and if said results are the same, resuming execution of said application program on the microprocessors.

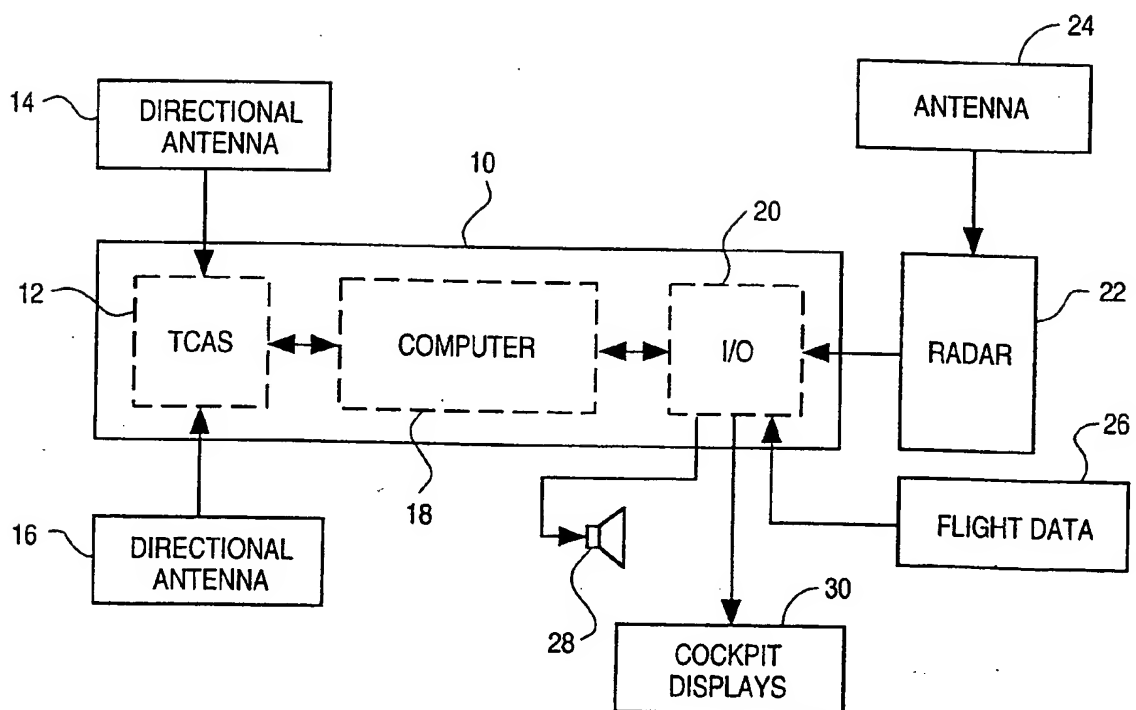
20 8. The method of Claim 7 additionally including the step of voting after said comparison of the result of said application program to select the result that are the same and resuming execution of said execution of said application program only on the microprocessors that produced the selected result.

9. The method of Claim 7 wherein after said comparison including the step of
25 outputting the result of the application program from the microprocessor to another system.

10. The method of Claim 9 including voting on the results of said application program to select the result from at least two microprocessors that are the same for output to said other system.

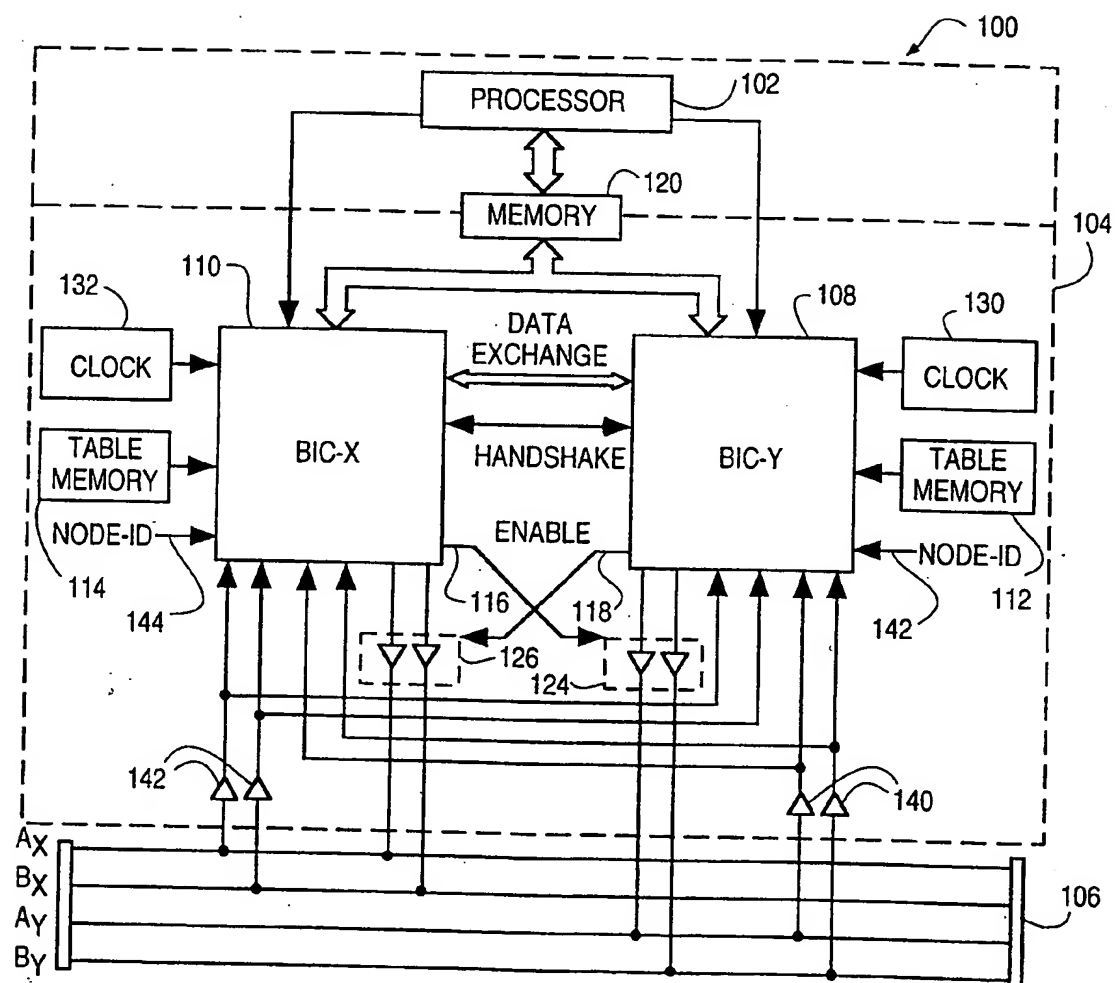
1/15

Fig. 1



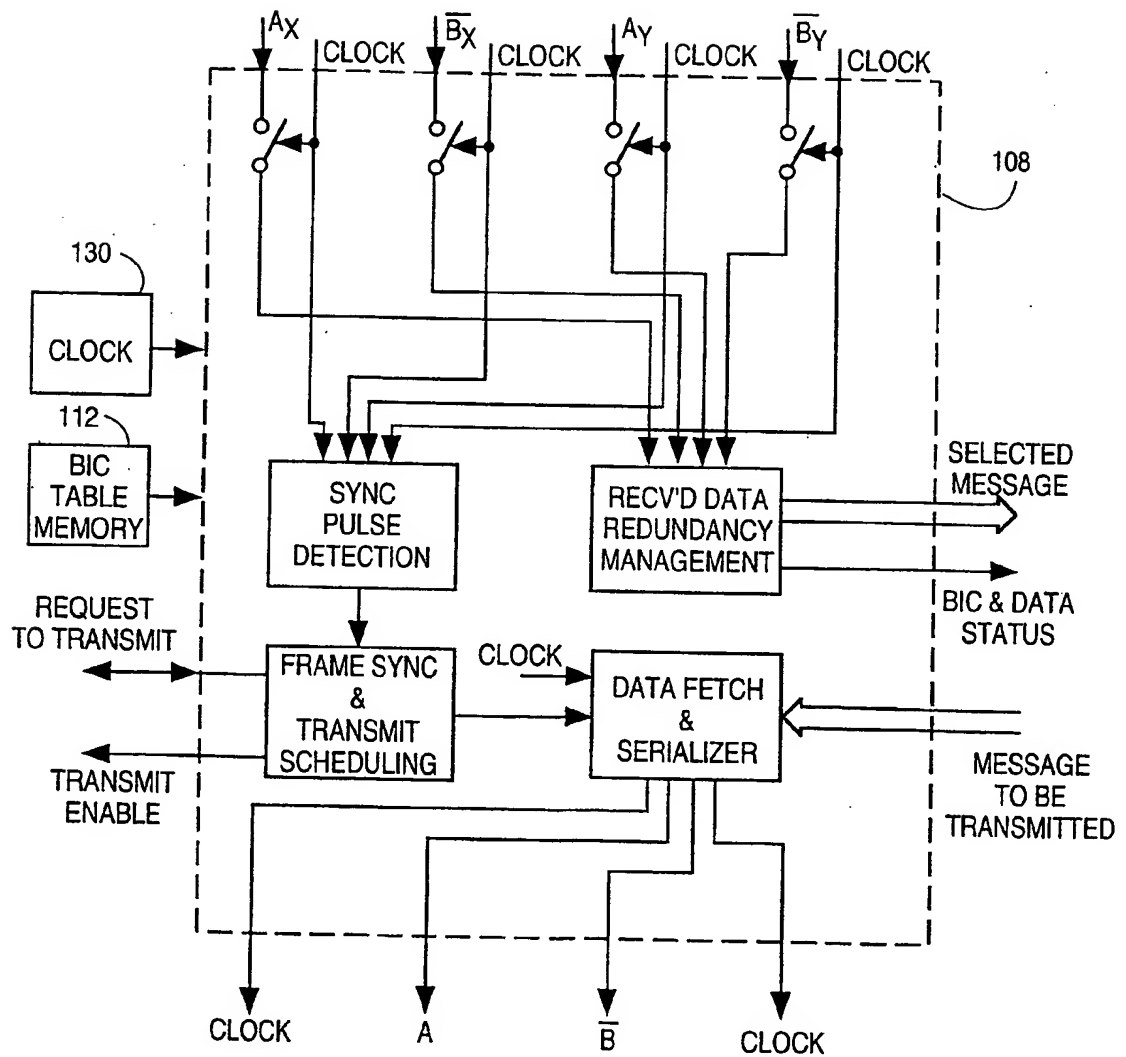
2/15

Fig. 2



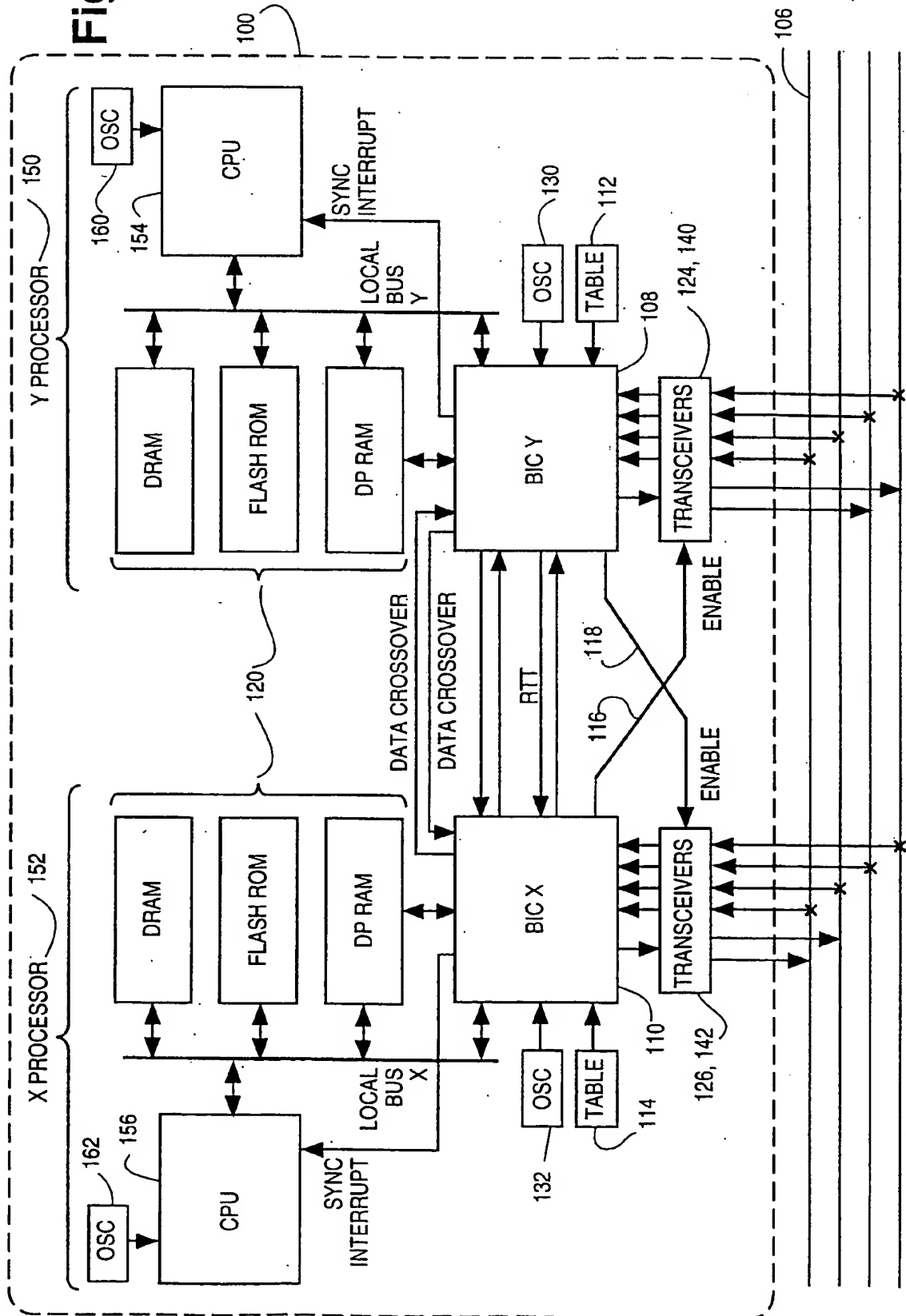
3/15

Fig. 3



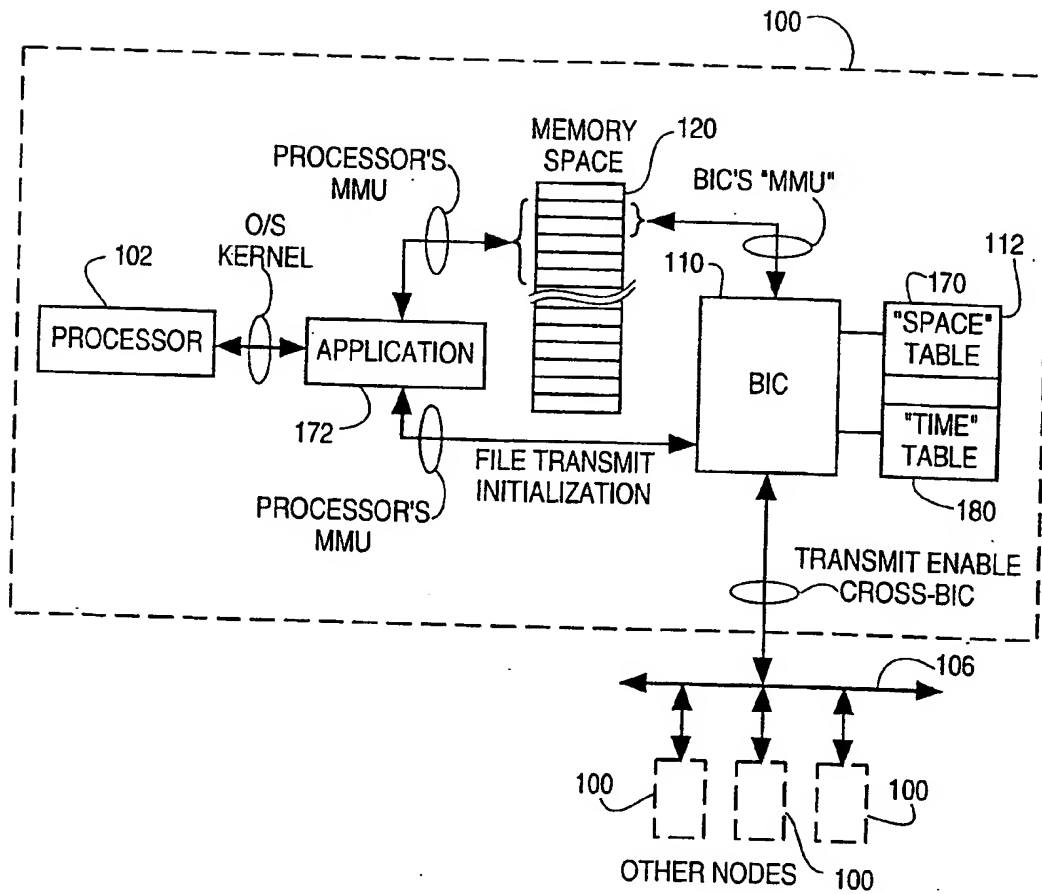
4/15

Fig. 4



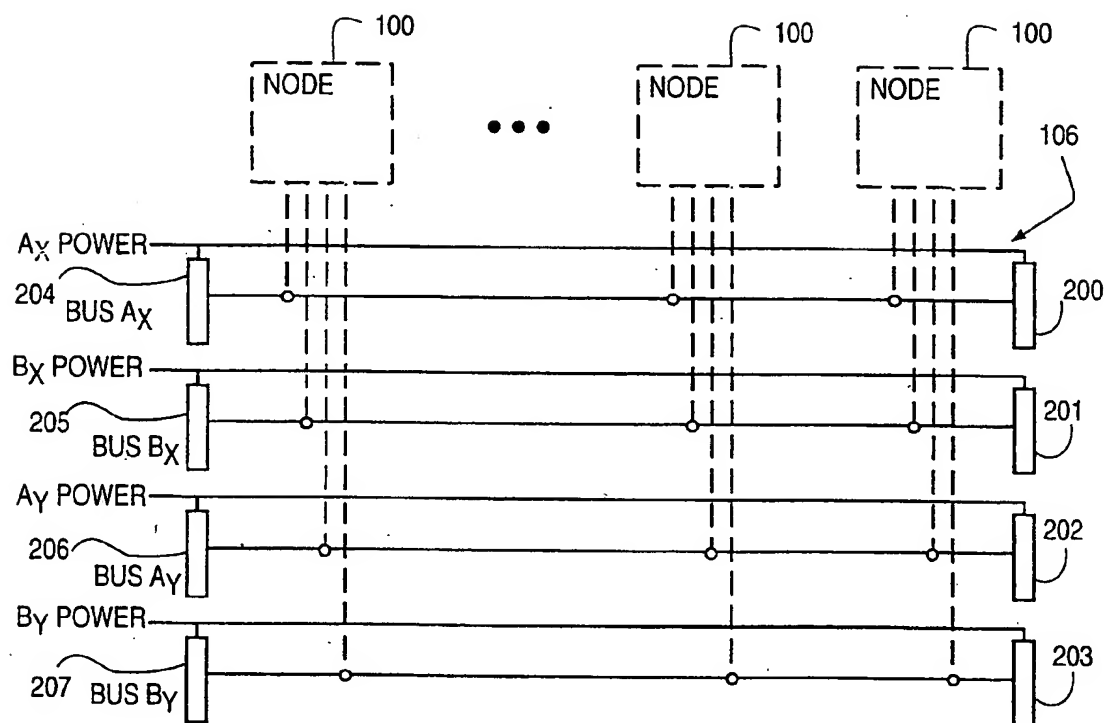
5/15

Fig. 5



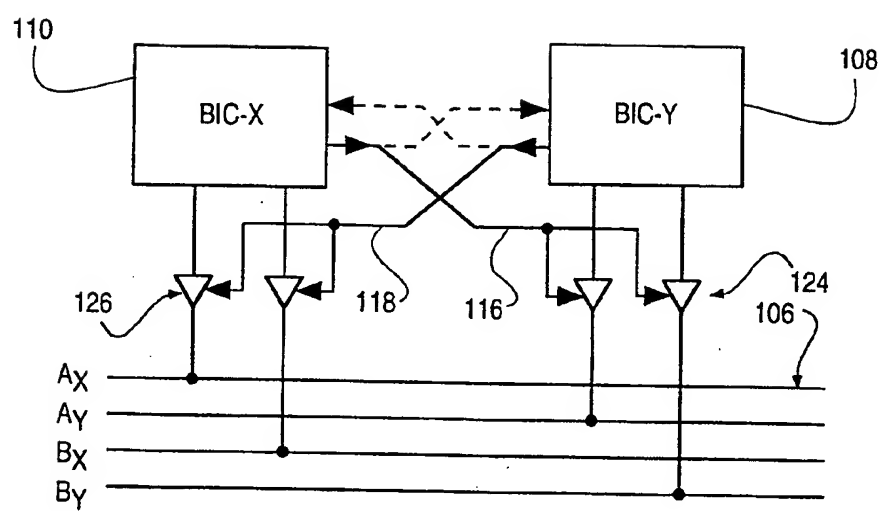
6/15

Fig. 6



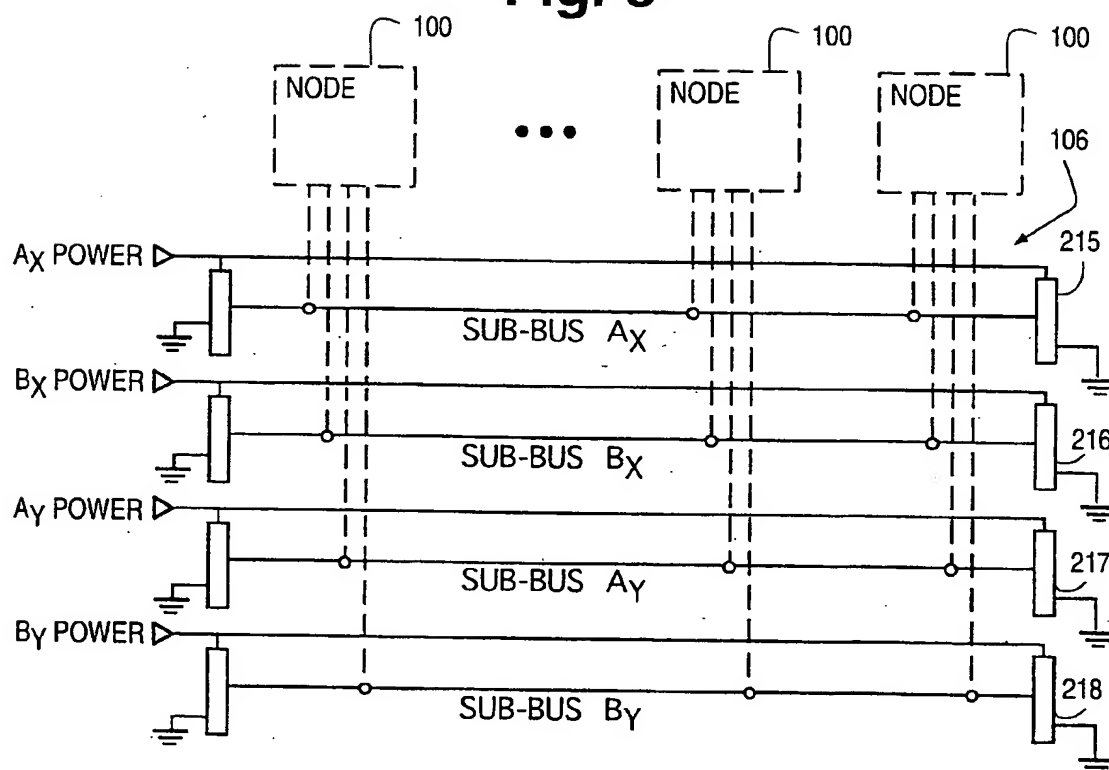
7/15

Fig. 7

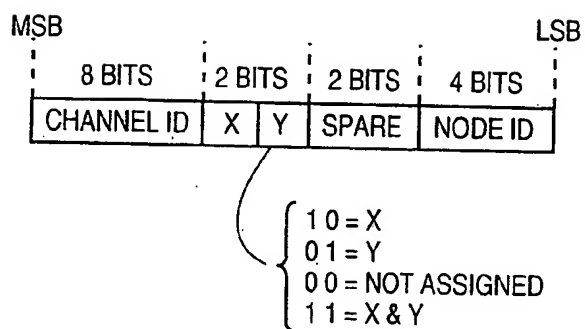


8/15

Fig. 8

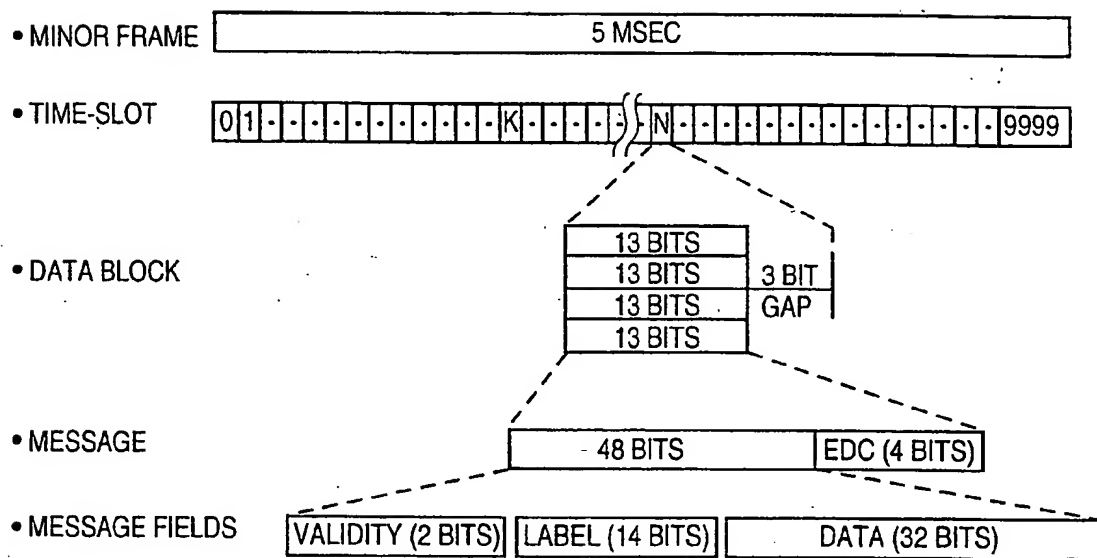


9/15

Fig. 9

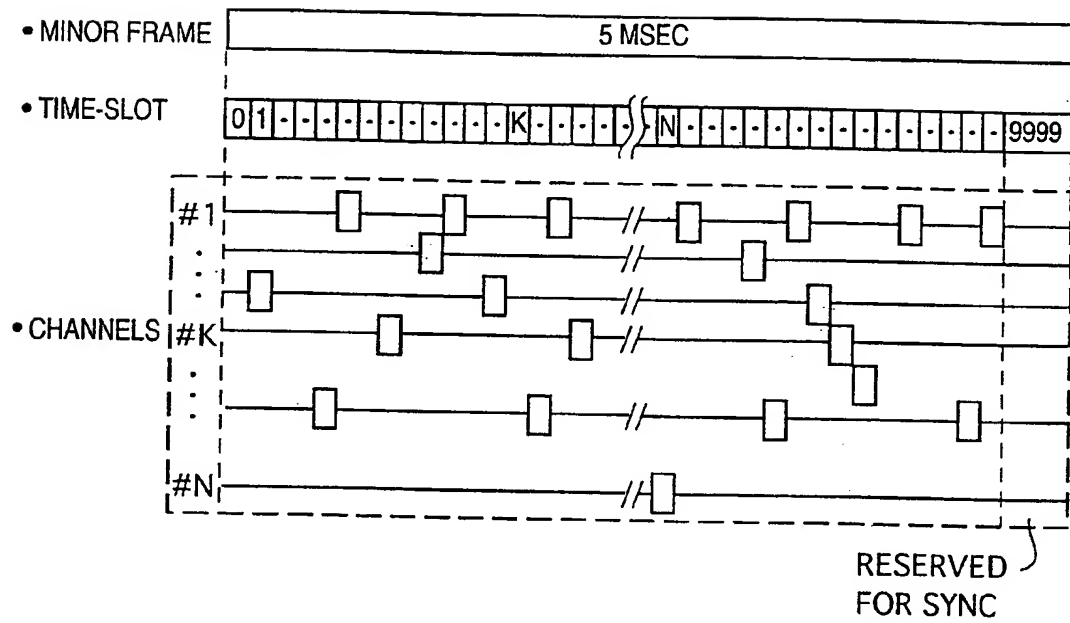
10/15

Fig. 10



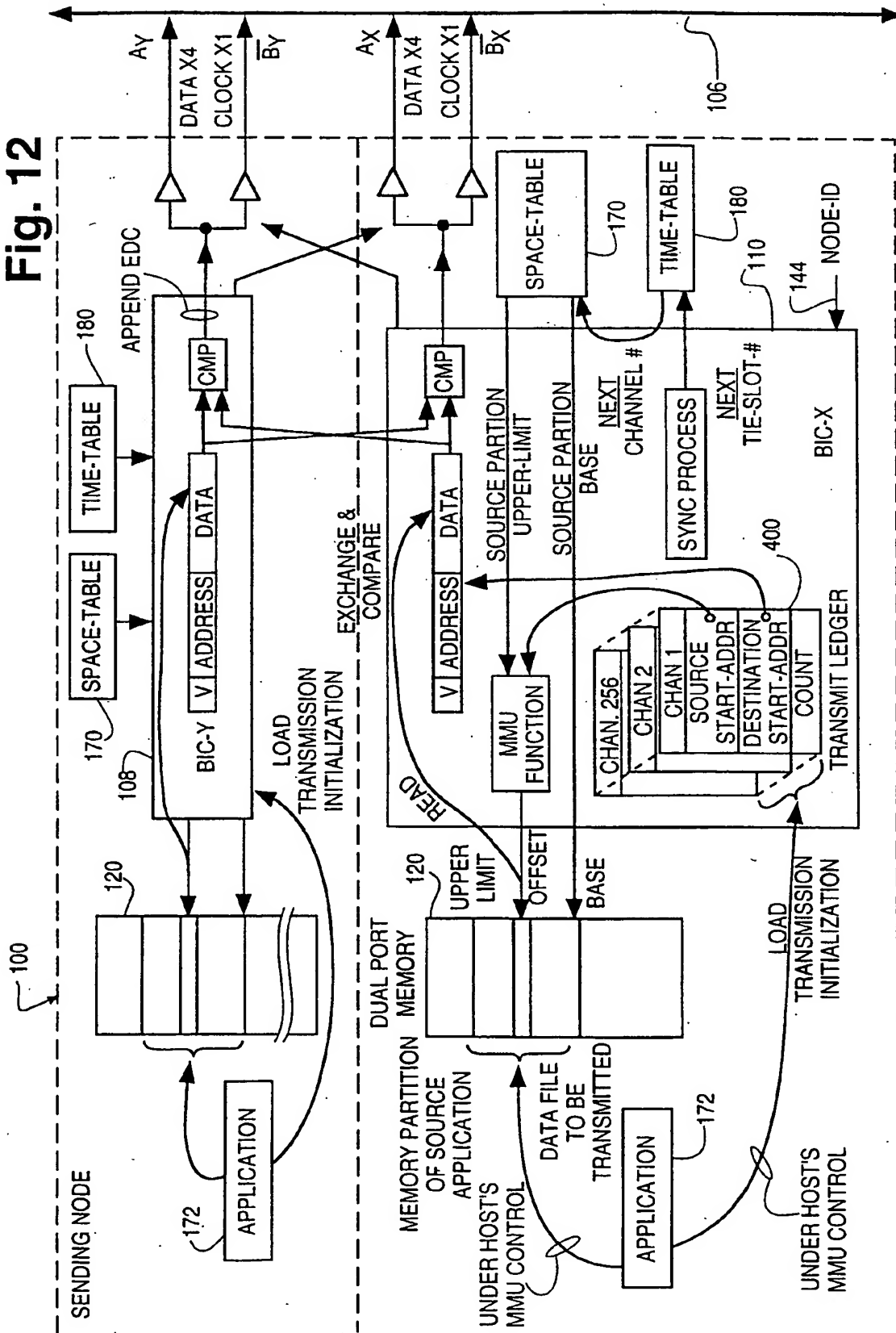
11/15

Fig. 11



12/15

Fig. 12



13/15

Fig. 13

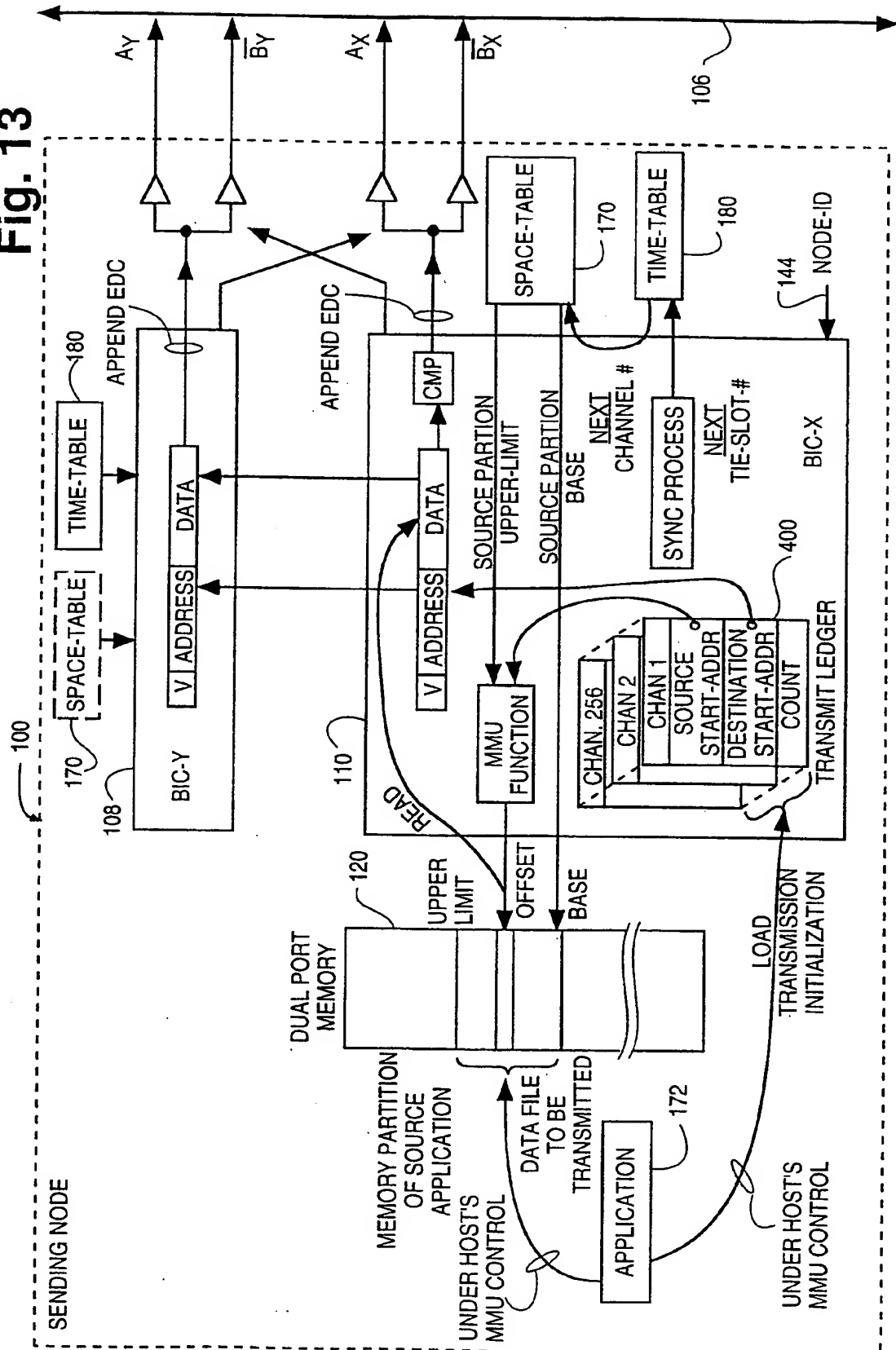


Fig. 14

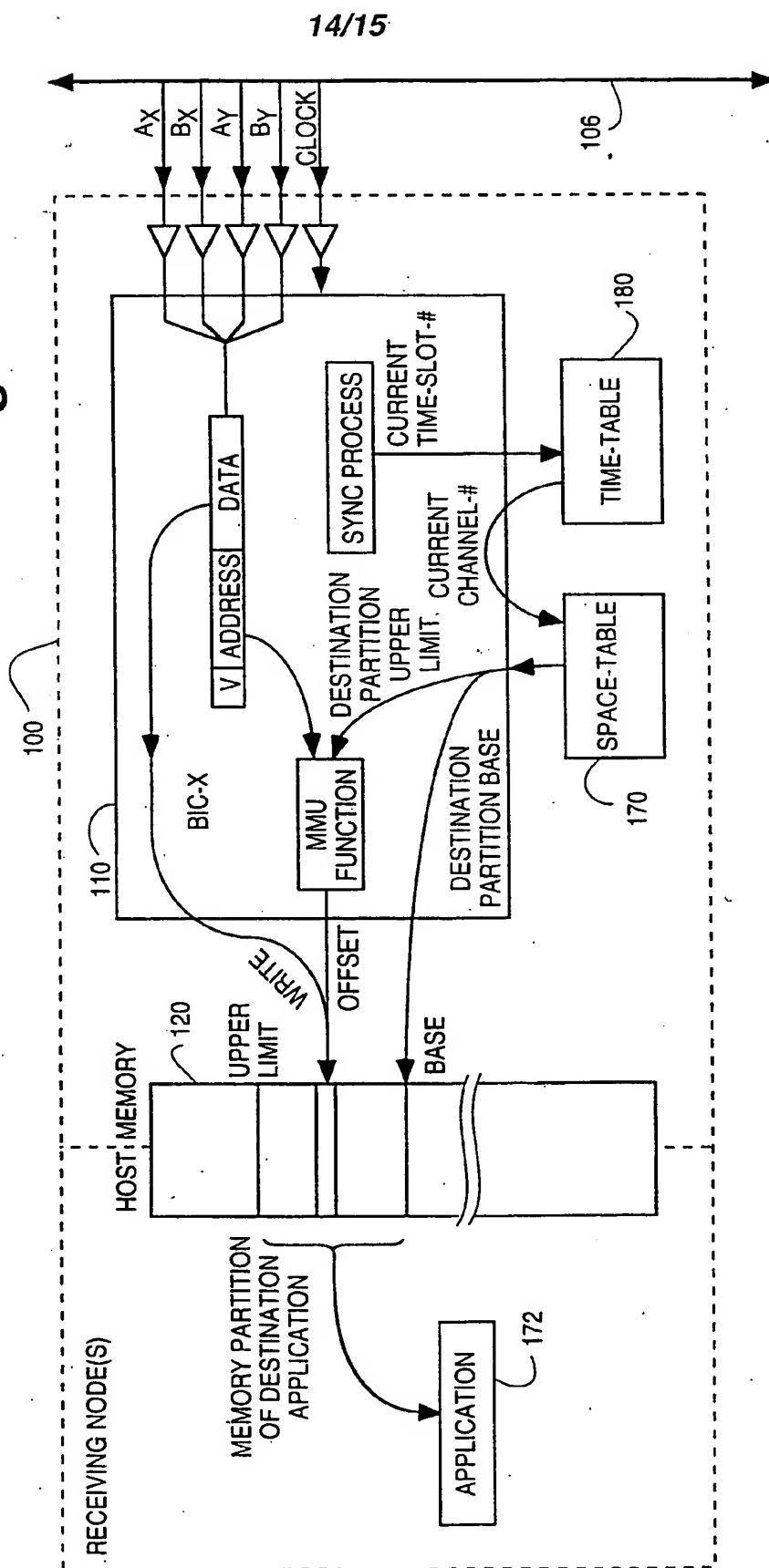


Fig. 15

